CSC 551: Web Programming

Spring 2004

Java Applets

- Java vs. C++
  - instance vs. class variables, primitive vs. reference types, inheritance
- graphical applets
  - Graphics object: drawString, drawLine, drawOval, drawRect, …
  - double buffering
- GUI applets
  - GUI elements, layout, event handling

Java vs. C++

Java syntax borrows from C++ (and C)

- primitive types: same as C++, but sizes are set

  byte (8 bits) char (16 bits) short (16 bits) int (32 bits) long (64 bits)
  float (32 bits) double (64 bits) boolean
- variables, assignments, arithmetic & relational operators: same as C++
- control structures: same as C++, but no goto
- functions: similar to C++, but must belong to class & must specify public/private

in Java, every variable & method belongs to a class

- as in C++, by default each object has its own copies of data fields

  thus, known as *instance variables*
- as in C++, a variables declared static are shared by all class objects

  thus, known as *class variables*
- similarly, can have a static method (*class method*)

  can only operate on class variables, accessible from the class itself

```
class Math
{
  public static final double PI = 3.14159;             // access as Math.PI
  public static double sqrt(double num) { . . . }      // access as in Math.sqrt(9.0)
  . . .
}
```

Primitive vs. reference types

primitive types are handled exactly as in C++

- space for a primitive object is implicitly allocated
  - → variable refers to the actual data (stored on the stack)

reference types (classes) are handled differently

- space for a reference object must be explicitly allocated using `new`
  - → variable refers to a pointer to the data (which is stored in the heap)

Note: unlike with C++, programmer is not responsible for deleting dynamic objects
JVM performs automatic garbage collection to reclaim unused memory

Java only provides by-value parameter passing

- but reference objects are implemented as pointers to dynamic memory
- resulting behavior mimics by-reference

```
public void Init(int[] nums)
{
    for (int i = 0; i < nums.length; i++) {
        nums[i] = 0;
    }
}
_______________________________________________
int nums[] = new int[10];
Init(nums);
```

Java libraries

String class (automatically loaded from `java.lang`)

```
int length()
char charAt(index)
int indexOf(substring)
String substring(start, end)
String toUpperCase()
boolean equals(Object)
...
```

```
String str = "foo"
String str = new String("foo");
```

Array class (automatically loaded from `java.lang`)

```
int length           instance variable
Type [](index)       operator
String toString()
...
```

```
int[] nums = {1,2,3,4,5};
int[] nums = new int[10];
```

Java provides extensive libraries of data structures & algorithms

```
java.util →       Date          Random        Timer
                  ArrayList     LinkedList    Stack
                  TreeSet       HashSet
                  TreeMap       HashMap
```

Applet behavior

recall

- the init method is called when the applet is first loaded
  useful for initializing variables & objects

- the paint method is called immediately after init, and whenever the applet needs to be redrawn (e.g., after window resized or obscured)

when paint is called, it is given the default Graphics object

- Graphics methods include

  `void drawString(String msg, int x, int y)`

  `void setColor(Color color)`

  `Color` class is predefined, constants include:
  `Color.red, Color.blue, Color.black, ...`

Hello again

```
import java.awt.*;
import java.applet.*;
import java.util.Random;

/**
 * This class displays lots of "Hello world!"s on the applet window.
 */
public class HelloWorld2 extends Applet
{
    private static final int NUM_WORDS=100;
    private static final Color[] colors =
          {Color.black,Color.red,Color.blue,Color.green,Color.yellow};
    private static Random randy;

    private int RandomInRange(int low, int high)
    {
          return (Math.abs(randy.nextInt()) % (high-low+1)) + low;
    }

    public void init()
    {
          randy = new Random();
    }

    public void paint(Graphics g)
    {
          for (int i = 0; i < NUM_WORDS; i++) {
                    int x = RandomInRange(1, 140);
                    int y = RandomInRange(10, 200);
                    g.setColor(colors[RandomInRange(0,4)]);
                    g.drawString("Hello world!", x, y);
          }
    }
}
```

store colors in an array
- pick random index and change color using `setColor`

Random class provides methods for generating random values

override init method to allocate & initialize (similar to a constructor)

```
<applet code="HelloWorld2.class" height=200 width=200>
You must use a Java-enabled browser to view this applet.
</applet>
```

view page in browser

Parameters & applet dimensions

recall:

- can specify parameters in the HTML document using `<PARAM>` tags
- access the parameter values (based on name) using `getParameter` method

can also access the dimensions of an applet using a Dimension object

```
Dimension dim = getSize();    // stores applet dimensions
```

can then access applet height via `dim.height`

can then access applet width via `dim.width`

Adaptive hello

```
import java.awt.*;
import java.applet.*;
import java.util.Random;

/**
 * This class displays lots of "Hello world!"s on the applet window.
 */
public class HelloWorld3 extends Applet
{
    private static final Color[] colors =
          {Color.black,Color.red,Color.blue,Color.green,Color.yellow};
    private static Random randy;
    private Dimension dim;
    private int numReps;

    private int RandomInRange(int low, int high)
    {
          return (Math.abs(randy.nextInt()) % (high-low+1)) + low;
    }

    public void init()
    {
          randy = new Random();
          dim = getSize();
          numReps = Integer.parseInt(getParameter("reps"));
    }

    public void paint(Graphics g)
    {
          for (int i = 0; i < numReps; i++) {
              int x = RandomInRange(1, dim.width-60);
              int y = RandomInRange(10, dim.height);
              g.setColor(colors[RandomInRange(0,4)]);
              g.drawString("Hello world!", x, y);
          }
    }
}
```

`getParameter` accesses the values of the parameters

here, specify number of reps in Web page

uses `getSize` to get dimensions, pick random coords for text within the applet

```
<applet code="HelloWorld3.class" height=300 width=400>
  <param name="reps" value=200>
  You must use a Java-enabled browser to view this applet.
</applet>
```

view page in browser

Applet graphics

in addition to displaying text

- can also draw figures on a Graphics object

```
void drawLine(int x1, int y1, int x2, int y2)

void drawRect(int x, int y, int width, int height)
void fillRect(int x, int y, int width, int height)

void drawOval(int x, int y, int width, int height)
void fillOval(int x, int y, int width, int height)
```

EXAMPLE: draw a red circle inscribed in a square, then draw random dots (dart pricks)

- by counting the number of dots inside vs. outside the circle, can estimate the value of π

π = 4 * (area of circle/area of square)

Graphical applet

```
public class Monte1 extends Applet
{
   private static Random randy;
   private int NUM_POINTS;
   private int SIZE;

   private int RandomInRange(int low, int high) { CODE OMITTED }
   private double distance(int x1, int y1, int x2, int y2) { CODE OMITTED }

   public void init()
   {
       randy = new Random();
       NUM_POINTS = Integer.parseInt(getParameter("points"));
       Dimension dim = getSize();
       SIZE = Math.min(dim.width, dim.height);
   }

   public void paint(Graphics g)
   {
       g.setColor(Color.red);
       g.fillOval(0, 0, SIZE, SIZE);
       for (int i = 0; i < NUM_POINTS; i++) {
           int x = RandomInRange(0, SIZE);
           int y = RandomInRange(0, SIZE);
           if (distance(x, y, SIZE/2, SIZE/2) < SIZE/2) {
               g.setColor(Color.white);
           }
           else {
               g.setColor(Color.black);
           }
           g.drawLine(x, y, x, y);
       }
   }
}
```

`init` method creates random number generator & gets parameters

`paint` method draws a circle and a bunch of random points

```
<applet code="Monte1.class" height=300 width=300>
  <param name="points" value=20000>
  You must use a Java-enabled browser...
</applet>
```

view page in browser

Double buffering

note: paint is called every time the page is brought to the front

- in current version of Monte, this means new dots are drawn each time the page is obscured and then brought back to the front
  - → *wastes time redrawing*
  - → *dots are different each time the applet is redrawn*

the double buffering approach works by keeping an off-screen image

- in the init method (which is called when the page loads):
  draw the figures on a separate, off-screen Graphics object
- in the paint method (which is called whenever the page is brought forward):
  simply display the off-screen image on the screen

Buffered applet

```
public class Monte2 extends Applet
{
    . . .
    private Image offScreenImage;
    private Graphics offScreenGraphics;
    . . .
    public void init()
    {
        randy = new Random();
        NUM_POINTS = Integer.parseInt(getParameter("points"));
        Dimension dim = getSize();
        SIZE = Math.min(dim.width, dim.height);

        offScreenImage = createImage(SIZE, SIZE);
        offScreenGraphics = offScreenImage.getGraphics();

        offScreenGraphics.setColor(Color.red);
        offScreenGraphics.fillOval(0, 0, SIZE, SIZE);
        for (int i = 0; i < NUM_POINTS; i++) {
            int x = RandomInRange(0, SIZE);
            int y = RandomInRange(0, SIZE);
            if (distance(x, y, SIZE/2, SIZE/2) < SIZE/2) {
                offScreenGraphics.setColor(Color.white);
            }
            else {
                offScreenGraphics.setColor(Color.black);
            }
            offScreenGraphics.drawLine(x, y, x, y);
        }
    }

    public void paint(Graphics g)
    {
        g.drawImage(offScreenImage, 0, 0, null);
    }
}
```

`init` method is called when page is loaded

does drawing to a separate, off-screen Graphics object

`paint` is called after `init` and whenever the applet is revisited

Note: don't see image in progress

```
<applet code="Monte2.class" height=300 width=300>
  <param name="points" value=20000>
  You must use a Java-enabled browser...
</applet>
```

view page in browser

```
public class Monte3 extends Applet
{
    . . .

  public void init() {
    randy = new Random();
    NUM_POINTS = Integer.parseInt(getParameter("points"));
    Dimension dim = getSize();
    SIZE = Math.min(dim.width, dim.height);
  }

  public void paint(Graphics g) {
    if (offScreenImage == null) {
        offScreenImage = createImage(SIZE, SIZE);
        offScreenGraphics = offScreenImage.getGraphics();

        offScreenGraphics.setColor(Color.red);
        g.setColor(Color.red);
        offScreenGraphics.fillOval(0, 0, SIZE, SIZE);
        g.fillOval(0, 0, SIZE, SIZE);
        for (int i = 0; i < NUM_POINTS; i++) {
            int x = randomInRange(0, SIZE);
            int y = randomInRange(0, SIZE);
            if (distance(x, y, SIZE/2, SIZE/2) < SIZE/2) {
                offScreenGraphics.setColor(Color.white);
                g.setColor(Color.white);
            }
            else {
                offScreenGraphics.setColor(Color.black);
                g.setColor(Color.black);
            }
            offScreenGraphics.drawLine(x, y, x, y);
            g.drawLine(x, y, x, y);
        }
    }
    else {
        g.drawImage(offScreenImage, 0, 0, null);
    }
  }
}
```

Better buffering

if want to see image as it is drawn, must be done in `paint`

when first loaded, have `paint` draw on the graphics screen and also to an off-screen buffer

on subsequent repaints, simply redraw the contents of the off-screen buffer

```
<applet code="Monte3.class" height=300 width=300>
  <param name="points" value=20000>
</applet>
```

view page in browser

GUI elements in applets

Java has extensive library support for GUIs (Graphical User Interfaces)

- has elements corresponding to HTML buttons, text boxes, text areas, …

each element must be created and explicitly added to the applet

```
nameLabel = new Label("User's name");
add(nameLabel);

nameField = new TextField(20);
nameField.setValue("Dave Reed");
add(nameField);
```

Java provides several classes for controlling layout

- `FlowLayout` is default
- `BorderLayout` allows placement of elements around the borders of the applet
- a `Panel` can contain numerous elements

Text boxes

```
public class Monte4 extends Applet
{
  . . .

  private Label insideLabel;
  private TextField insideField;
  private Label outsideLabel;
  private TextField outsideField;

  public void init()
  {
    randy = new Random();
    NUM_POINTS =
    Integer.parseInt(getParameter("points"));
    Dimension dim = getSize();
    SIZE = Math.min(dim.width, dim.height);

    setLayout(new BorderLayout());

    Panel p = new Panel();
    insideLabel = new Label("Inside:");
    p.add(insideLabel);
    insideField = new TextField(5);
    p.add(insideField);
    outsideLabel = new Label("Outside:");
    p.add(outsideLabel);
    outsideField = new TextField(5);
    p.add(outsideField);

    add(p, BorderLayout.SOUTH);
  }
```

```
  public void paint(Graphics g)
  {
    . . .

    insideField.setText("0");
    outsideField.setText("0");

      . . .

      if (distance(x, y, SIZE/2, SIZE/2) < SIZE/2) {
        g.setColor(Color.white);
        int value =
          Integer.parseInt(insideField.getText())+1;
        insideField.setText(""+value);
      }
      else {
        g.setColor(Color.black);
        int value =
          Integer.parseInt(outsideField.getText())+1;
        outsideField.setText(""+value);
      }

    . . .
  }
}
```

```
<applet code="Monte4.class" height=335 width=300>
  <param name="points" value=20000>
</applet>
```

view page in browser

Event handling

in order to handle events (e.g., text changes, button clicks), can use the *event delegation model*

- must specify that the class implements the `ActionListener` interface

```
public class Monte6 extends Applet implements ActionListener
```

- each source of events must be registered within the applet

```
dotButton = new Button("Click to generate dots");
dotButton.addActionListener();
```

- must have an `actionPerformed` method to handle events

```
public void actionPerformed(ActionEvent e)
{
  if (e.getSource() == dotButton) {
    drawDots();
  }
}
```

ActionListener

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import java.util.Random;

public class Monte5 extends Applet
                    implements ActionListener
{
  . . .
  private Button dotButton;

  public void init()
  {
    randy = new Random();
    NUM_POINTS =
      Integer.parseInt(getParameter("points"));
    Dimension dim = getSize();
    SIZE = dim.width;

    setLayout(new BorderLayout());
    dotButton =
      new Button("Click to generate dots");
    dotButton.addActionListener(this);
    add(dotButton, BorderLayout.SOUTH);

    drawCircle();
  }
```

```
public void drawCircle()
{
   CODE FOR DRAWING CIRCLE
}

public void drawDots()
{
  drawCircle();

  Graphics g = getGraphics();
  for (int i = 0; i < NUM_POINTS; i++) {
      CODE FOR DRAWING DOTS
  }
}

public void paint(Graphics g)
{
  g.drawImage(offScreenImage, 0, 0, null);
}

public void actionPerformed(ActionEvent e)
{
  if (e.getSource() == dotButton) {
    drawDots();
  }
}
}
```

```
<applet code="Monte5.class" height=325 width=300>
  <param name="points" value=20000>
</applet>
```

view page in browser

Applet examples

The Java Boutique has lots of sample applets with source code

- Graphing Calculator
- Mandlebrot Set
- Email front-end
- Web search front-end
- Java Tetris